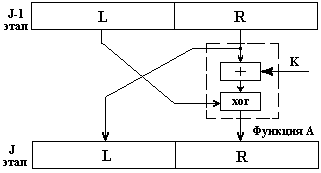
Шифр ГОСТ 28147-89

Общая схема на сети Файстеля:



Для того чтобы идея сетей Файстеля была окончательна ясна, рассмотрим простейший случай изображенный на рис. 1, где в функции А – выступит операции “mod 2” (“xor”), но это простейший случай, в более серьезной ситуации, например сокрытие информации государственной важности функция А может быть более сложной (сколько я видел функция А действительно бывает очень сложной):

Исходные данные:

L = 1110b, R = 0101, K = 1111b

Цель:

Получить шифрограмму

Решение:

1. (R + K) mod 24 = Smod, Smod = 0100b

2. (Smod + L) mod 2 = Sxor, Sxor = 1010b

3. L = R, R = Sxor

Итог:

 L = 0101b, R = 1010b

            Поясним наши действия:

1.       Эта операция сложение по mod 24. На практике такая операция сводится к простому сложению, где мы должны сложить два числа и проигнорировать перенос в 5й разряд. Так как, если проставить над разрядами двоичного представления числа проставить показатели степени, над пятым разрядом как раз будет показатель четыре, взглянем на рисунок ниже, где изображены действия нашей операции:

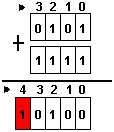


Рис. 2

Здесь я стрелкой указал на показатели степени, как видно, результат должен был получиться 10100, но так как при операции mod 24 игнорируется перенос, мы получаем 0100.

2. Эта операция в литературе называется mod 2, на языке ассемблера реализуется командой XOR. Но ее более правильное название mod 21. Без этой уникальной операции вряд ли можно построить быстрый, легко реализуемый алгоритм шифрования и при этом, чтобы он был еще довольно криптостойким. Уникальность этой операции заключается в том, что она сама себе обратная! К примеру, если число А поXORить с числом Б, в результате получим В, в дальнейшем достаточно переXORить числа Б и В между собой, чтобы получить прежнее значение А!

В этой операции мы получили 1010 имея числа 1110 и 0100, чтобы получить обратно 1110, достаточно переXORрить между собой числа 0100 и 1010! Более подробно об этой операции можно почитать в статье, которая вложена на сайте [www.wasm.ru](https://wasm.in/index.html), «Элементарное руководство по CRC\_алгоритмам обнаружения ошибок» автор, которой Ross N. Williams. В этом труде есть пункт - «5. Двоичная арифметика без учета переносов». Вот именно в этой статье и описана операция xor! Я восклицаю потому что в этой статье эта операция так расписана, что читатель не просто понимает как работает эта операция, он даже начинает ее видеть, слышать и чувствовать!

3. Это действие необходимо, чтобы при расшифровывании из шифрограммы можно было получить исходные значения.

2.2 Блочный шифр ГОСТ 28147-89

         Алгоритм шифрования ГОСТ 28147 – 89 относится к разряду блочных шифров работающих по архитектуре сбалансированных сетей Файстеля, где две части выбранного блока информации имеют равный размер. Алгоритм был разработан в недрах восьмого отдела КГБ преобразованного ныне в ФАПСИ и был закреплен, как стандарт шифрования Российской Федерации еще в 1989 году при СССР.

            Для работы данного метода алгоритма необходимо разбить информацию на блоки размером в 64 бита. Сгенерировать или ввести в систему шифрования, следующую ключевую информацию: ключ и таблицу замен. К выбору ключа и таблицы замен при шифровании следует отнестись очень серьезно, т.к. именно это фундамент безопасности вашей информации. О том, какие требования налагаются на ключ, и таблицу замен смотри пункт «Требования к ключевой информации».

При рассмотрении метода мы не будем заострять на этом внимания, т.к. эта статья, как я уже говорил выше, написана с целью, научить читающего, шифровать данные по методу простой замены данного алгоритма шифрования, но мы  обязательно коснемся этого вопроса в конце статьи.

Теоретический минимум.

3.1 Ключевая информация

Как я уже говорил выше, в шифровании данных активное участие принимают:

3.1.1. Ключ – это последовательность восьми элементов размером в 32 бита каждый. Далее будем обозначать символом К, а элементы из которых он состоит – k1,k2,k3,k4,k5,k6,k7,k8.

            3.1.2 Таблица замен – матрица из восьми строк  и шестнадцати столбцов, в дальнейшем – Hij. Каждый элемент на пересечении строки i и столбца j занимает 4 бита.

3.2 Основной шаг криптопреобразования

            Основным действием в процессе шифрования является – основной шаг криптопреобразования. Это ничто иное, как действие по шифрованию данных по определенному алгоритму, только название разработчики ввели уж больно громоздкое :smile3:.

            Прежде чем начать шифровать, блок разбивают на две части L и R, по 32 бита каждая. Выбирают элемент ключа и только потом подают эти две части блока, элемент ключа таблицу замен в функцию основного шага, результат основного шага это одна итерация базового цикла, о котором речь пойдет в следующем пункте. Основной шаг состоит из следующих действий:

Сложение часть блока R суммируется с элементом ключа K по mod 232. О подобной операции я описал выше, здесь тоже самое только показатель степени не «4», а «32» - результат этой операции в дальнейшем буду обозначать Smod.

Полученный ранее результат Smod делим на четырех битные элементы s7,s6,s5,s4,s3,s2,s1,s0 и подаем в функцию замены. Замена происходит следующим образом: выбирается элемент Smod - si, с начала начинаем с младшего элемента, и заменяем значением из таблицы замен по i - той строке и столбцу, на который указывает значение элемента si. Переходим к si+1 элементу и поступаем аналогичным образом и продолжаем так, пока не заменим значение последнего элемента Smod – результат этой операции будем обозначать как, Ssimple.

В этой операции значение Ssimple сдвигаем циклически влево на 11 бит и получаем Srol.

Выбираем вторую часть блока L и складываем по mod 2 с Srol, в итоге имеем Sxor.

На этой стадии часть блока L становится равным значению части R, а часть R в свою очередь инициализируется результатом Sxor и на этом функция основного шага завершена!

3.3 Базовые циклы: “32-З”, “32-Р”.

            Для того чтобы зашифровать информацию надо разбить ее на блоки размером в 64 бита, естественно последний блок может быть меньше 64 битов. Этот факт является ахиллесовой пятой данного метода «простая замена».  Так как его дополнение до 64 бит является очень важной задачей по увеличению криптостойкости шифрограммы и к этому чувствительному месту, если оно присутствует в массиве информации, а его может и не быть (к примеру, файл размером в  512 байт!), следует отнестись с большой ответственностью!

            После того как вы разбили информацию на блоки, следует разбить ключ на элементы:

K = k1,k2,k3,k4,k5,k6,k7,k8

            Само шифрование заключается в использовании, так называемых – базовых циклов. Которые в свою очередь включают в себя n – ое количество основных шагов криптопреобразования.

            Базовые циклы имеют, как бы это сказать, маркировку: n – m. Где n – количество основных шагов криптопреобразования в базовом цикле, а m – это «тип» базового цикла, т.е. о чем идет речь, о «З» ашифровывании или «Р» асшифровывании данных.

            Базовый цикл шифрования 32–З состоит из 32-х основных шагов криптопреобразования. В функцию реализующую действия шага подают блок N и элемент ключа К причем, первый шаг происходит с к1, второй над полученным результатом с элементом к2 и т.д. по следующей схеме:

k1,k2,k3,k4,k5,k6,k7,k8,k1,k2,k3,k4,k5,k6,k7,k8,k1,k2,k3,k4,k5,k6,k7,k8k8,k7,k6,k5,k4,k3,k2,k1

            Процесс расшифровывания 32–Р происходит аналогичным образом, но элементы ключа подаются в обратной последовательности:

k1,k2,k3,k4,k5,k6,k7,k8,k8,k7,k6,k5,k4,k3,k2,k1,k8,k7,k6,k5,k4,k3,k2,k1,k8,k7,k6,k5,k4,k3,k2,k1

Практика.

4.1 Реализация основного шага криптопреобразования

         После того как мы познакомились с теорией о том, как шифровать информацию настало посмотреть, как же происходит шифрование на практике.

Исходные данные:

            Возьмем блок информации N = 0102030405060708h, здесь части L и R равны:

L = 01020304h, R =05060708h, возьмем ключ:

K = ‘as28zw37q8397342ui238e2twqm2ewp1’ (это ASCII – коды, для того, чтобы посмотреть шестнадцатеричное представление, можно открыть этот файл в режим просмотра в Total Commander нажав на клавишу «F3» и далее клавишу «3»). В этом ключе значения элементов будут:

k1 = ‘as28’, k2 = ‘zw37’, k3 = ‘q839’, k4 = ‘7342’

k5 = ‘ui23’, k6 = ‘8e2t’, k7 = ‘wqm2’, k8 = ‘ewp1’

Также возьмем следующую таблицу замен:

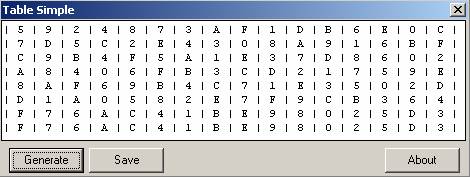


Рис. 3

Здесь строки нумеруются от 0 до 7, столбцы от 0 до F.

Предупреждение: Вся информация, в том числе и ключ с таблицей замен взята в качестве примера для рассмотрения алгоритма!

Задача:

Используя «Исходные данные», необходимо получить результат действия основного шага криптопреобразования.

Решение:

1. Выбираем часть R = 05060708h и элемент ключа k1 = ‘as28’, в шестнадцатеричном виде элемент ключа будет выглядеть так: 61733238h. Теперь же делаем операцию суммирования по mod 232:

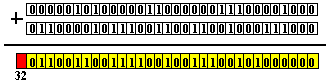


Рис. 4

Как видно на рисунке у нас не произошло переноса в 33 бит помеченный красным цветом и с показателем степени «32». А если бы у нас были бы другие значения R и элемента ключа – это вполне могло бы произойти, и тогда бы мы его проигнорировали, и в дальнейшем использовали только биты, помеченные желтым цветом.

         Такую операцию я выполняю командой ассемблера add:

            ; eax = R, ebx = ‘as28’

            add eax,ebx

            ; eax = Smod

Результат этой операции Smod = 66793940h

2. Теперь самая заковыристая операция, но если присмотреться по внимательней, то она уже не такая страшная, как кажется в первое время. Представим Smod в следующем виде:

https://wasm.in/archive/pub/25/pic/gost29147-89/image005.gif

Рис. 5

Я постарался наглядно представить элементы Smod на рисунке, но все равно поясню:

s0 = 0, s1 = 4, s2 = 9 и т.д.

Теперь начиная с младшего элемента s0, производим замену. Вспоминая пункт «3.2 Основной шаг криптопреобразования» i ­– строка, si – столбец, ищем в нулевой строке и нулевом столбце значение:

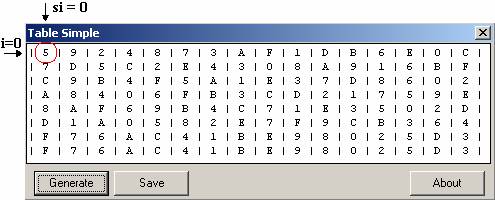


Рис.6

Таким образом, текущее значение Smod, не 66793940h, а 66793945h.

Приступаем заменять s1, т.е. четверку. Используя первую строку и четвертый столбец (s1= 4!). Глядим на рисунок:

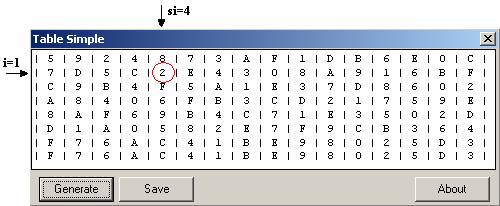


Рис. 7

Теперь уже значение Smod, не 66793945h, 66793925h. Я предполагаю, что теперь алгоритм замены читателю понятен, и я могу сказать, что после конечный результат Ssimple будет иметь следующее значение – 11e10325h.

О том, как это проще всего реализовать в виде команд ассемблера я расскажу позже в следующем пункте, после того, как расскажу о расширенной таблице.

Полученное значение Ssimple мы должны сдвинуть на 11 бит влево.

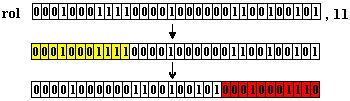


Рис. 8

Как видно это действие довольно простое, и реализуется одной командой языка ассемблера – rol  и результат этой операции Srol равен 0819288Fh.

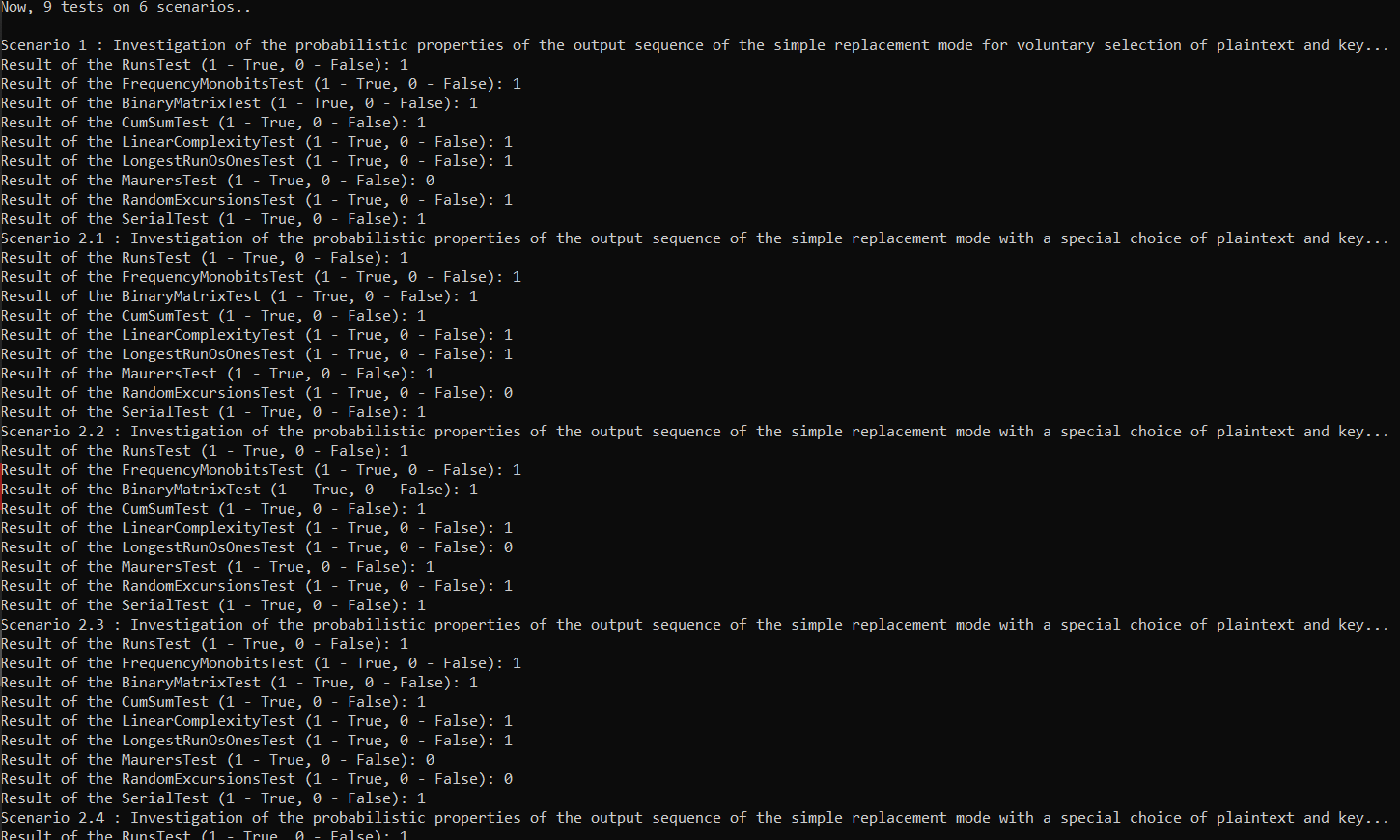
4. Теперь же остается часть L нашего блока информации поXORить со значением Srol. Я беру калькулятор от w2k sp4 и получаю Sxor = 091b2b8bh.

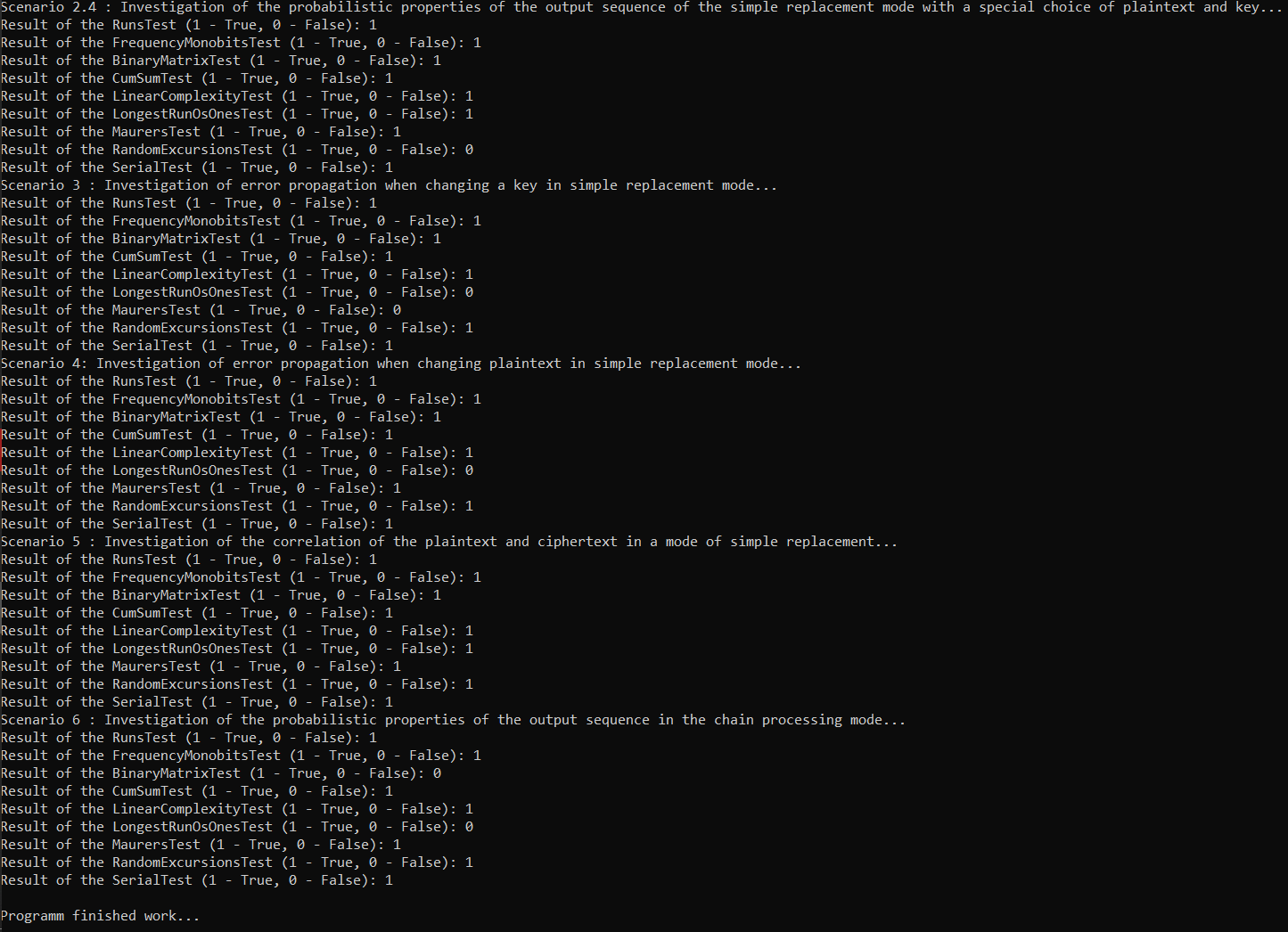
5. Это действие итоговое и мы просто присваиваем, чисти R значение части L, а часть L инициализируем значением Sxor.

Конечный результат:

L = 091b2b8bh, R = 01020304h

Также проводилось тестирование программы. Результат работы программы:





Итоговая таблица с тестами:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Сценарий 1 | Сценарий 2.1 | Сценарий 2.2 | Сценарий 2.3 | Сценарий 2.4 | Сценарий 3 | Сценарий 4 | Сценарий 5 | Сценарий 6 |
| RunsTest | True | True | True | True | True | True | True | True | True |
| FrequencyMonobitsTest | True | True | True | True | True | True | True | True | True |
| BinaryMatrixTest | True | True | True | True | True | True | True | True | False |
| CumSumTest | True | True | True | True | True | True | True | True | True |
| LinearComplexityTest | True | True | True | True | True | True | True | True | True |
| LongestRunOsOnesTest | True | True | False | True | True | False | False | True | False |
| MaurersTest | False | True | True | False | True | False | True | True | True |
| RandomExcursionTest | True | False | True | False | False | True | True | True | True |
| SerialTest | True | True | True | True | True | True | True | True | True |

Можно наблюдать, что достаточно много тестов пройдены не были, что говорит о том, что алгоритм шифрования ГОСТ 28147-89 имеет некоторые слабости и уступает современным криптосистемам. Можно сказать, что ГОСТ устойчив к таким широко применяемым методам, как линейный и дифференциальный криптоанализ. Обратный порядок использования ключей в последних восьми раундах обеспечивает защиту от атак скольжения и отражения.